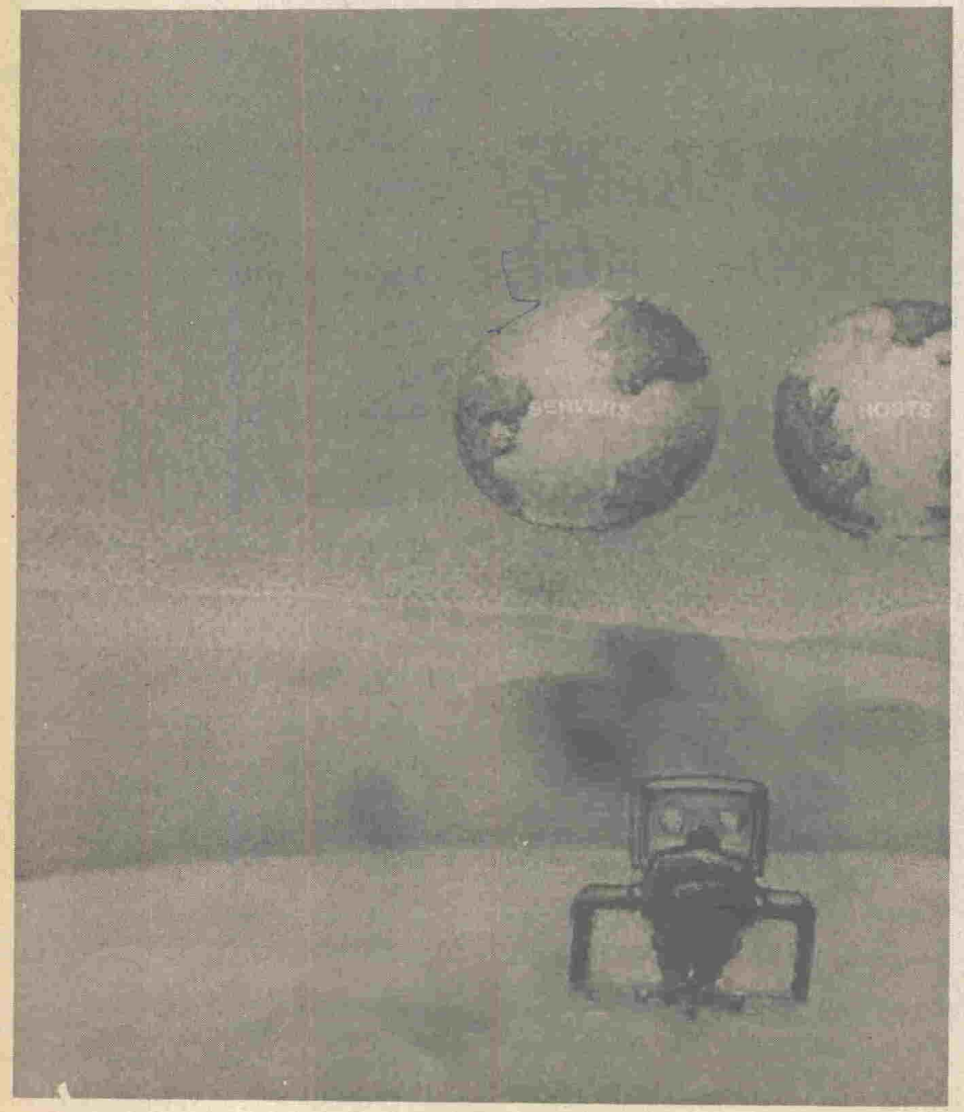
CONNECTIVITY
ผูกใจคอมพิวเตอร์เป็นหนึ่งเดียว

ทำความเข้าใจกับคำว่า CONNECTIVITY โดยวิธีอุปมาและอุปไมยอย่างง่าย ๆ



ผู้บริหารกับนักวิชาการทางคอมพิวเตอร์มักจะมีความเข้าใจไม่ตรงกันอยู่เรื่องหนึ่ง คือ เมื่อเราพูดว่าเราสามารถเชื่อมโยงเครื่องคอมพิวเตอร์เข้าหากันได้แล้ว ผู้บริหารก็จะสรุปเลยว่า ผู้ใช้บนเครื่องทั้งสองสามารถเรียกใช้ข้อมูลของกันและกันได้เลย แต่ในสายตาของนักวิชาการคอมพิวเตอร์นั้นจะบอกได้เลยว่ามันยังอยู่ห่างไกลจากความเป็นจริงอีกหลายโค้ง ความเข้าใจผิดกันนี้จะเกิดผลอย่างรุนแรงเมื่อผู้พูด (ว่า...เราเชื่อมเครื่องกันได้) เป็นแค่เซลส์แมนหรือเซลส์วูแม่น ทั้งนี้เพราะผู้ขายเหล่านี้มักจะพยายามพูดในทางบวกมากกว่าลบ และมักจะแถมท้ายเป็นทำนองดูถูกและเหยียดหยามฝีมือของโปรแกรมเมอร์ว่าไม่รู้เรื่อง ผู้บริหารทั้งหลายฟังแล้วเคลิ้ม แล้วมาเรียกร้องและเร่งรัดเอากับโปรแกรมเมอร์ของตน ปัญหานี้กำลังทยอยเกิดขึ้นในองค์การขนาดใหญ่ที่มีคอมพิวเตอร์หลาย ๆ เครื่อง และกำลังจะเป็นเรื่องฮิตที่สุดใน 2-3 ปีข้างหน้า ทั้งนี้เพราะหน่วยงานต่าง ๆ มีความต้องการที่จะเชื่อมโยงเครื่องคอมพิวเตอร์ของตนกับเครื่องคอมพิวเตอร์ของคนอื่น วันนี้ผมจึงคิดว่าเราน่าจะมาหาทางอธิบายให้เข้าใจกันเสียก่อน ผมจะพยายามใช้วิธีอุปมาและอุปไมย เพื่อให้ท่านที่เป็นผู้บริหารพอจะฟังกันเข้าใจได้

CONNECTIVITY ผูกใจคอมพิวเตอร์เป็นหนึ่งเดียว

อันดับแรกของการเชื่อมโยง

สิ่งแรกที่ผมอยากจะให้เข้าใจตรงกันก็คือ คำว่า "เชื่อมโยงกันได้" นั้นมี 2 ระดับ ระดับแรกเป็นการเอาฮาร์ดแวร์มาเชื่อมโยงกัน ความหมายก็คือ เราต้องมีสายสื่อสารเชื่อมจากตัวควบคุมการสื่อสารของเครื่องที่ 1 ไปยังตัวควบคุมการสื่อสารของเครื่องที่ 2 เรื่องนี้บางคนบอกว่าง่ายมาก แต่จริง ๆ แล้วมันก็ไม่ง่ายเท่าใดนัก เช่น ในประการแรกเราต้องใช้ปลั๊กสายที่เป็นระบบเดียวกัน ปลั๊กสายสื่อสารนั้นมีหลายสิบชนิด บางชนิดเป็นตัวกลม ๆ มีขา 5 ขา บางชนิดเป็นตัวแบน ๆ มีขา 17 ขา โดยแบ่งเป็น 2 แถว บางชนิดเป็นตัวแบน ๆ ยาวกว่าเก่าเล็กน้อย มีขา 30 กว่าขา และจัดแบ่งเป็น 3 แถว ขาใน 3 แถวนี้เยื้องกันโดยทำมุม 60 องศา หรือคือสลับฟันปลา แต่บางชนิดมีขาเรียงเหมือนแถวทหาร ปัญหาประการที่ 2 คือ ปลั๊กที่กล่าวมาเหล่านี้มีชนิดตัวผู้ (มีขายื่นออกมา) กับตัวเมีย (เป็นรูเพื่อเสียบเข้าไป) การเดินสายสัญญาณต่าง ๆ จะต้องตรงกันพอดี ใครไม่รู้กฎเกณฑ์ทางด้านช่างสื่อสารก็จะต่อสายไม่ถูก ถ้าเอาสายไฟบ้านหรือไฟดีซีใส่เข้าไปตรงกับสายสัญญาณดิจิตอลก็อาจทำให้วงจรไฟฟ้าด้านใดด้านหนึ่งพังได้

เรื่องนี้อาจใช้สำหรับย้อนถามพวกเซลส์แมนและเซลส์วูแม่นได้ คือใครบอกว่าทำได้ง่ายก็ลองให้เขาเปิดดูหลังเครื่องคอมพิวเตอร์แล้วชี้ให้ดูหน่อยว่าจะต้องต่อสายไหนเข้ากับสายไหน หรือตัดสายเคเบิลที่มีอยู่ระโยงระยางหลังเครื่องคอมพิวเตอร์ออกสัก 5-6 เส้น แล้วให้เขาลองต่อกลับเข้าไปใหม่ดู ถ้าคนพวกนี้บอกว่าเขาไม่ใช่ช่าง เราก็ลองบอกว่าเราจะตัดสายดู แล้วขอให้ไปหาช่างมาเชื่อมต่อให้เรา ผมบอกได้เลยว่าเขาหาไม่ได้ ทั้งนี้เพราะช่างที่รู้เรื่องจริง ๆ ในเมืองไทยนั้นมีไม่ถึง 100 คน ช่างส่วนใหญ่จะทำได้แค่เอาปลั๊กและสายเคเบิลสำเร็จรูปมาเชื่อมโยงกันเท่านั้น ปลั๊กและสายเคเบิลสำเร็จรูปนี้ผลิตโดยบริษัทผู้ค้าคอมพิวเตอร์ และมักจะทำไว้เพื่อเชื่อมโยงเครื่องยี่ห้อเดียวกัน ช่างประเภทหลังนี้ได้รับการฝึกอบรมจากบริษัทผู้ผลิตโดยเฉพาะและรู้เรื่องจำกัดสำหรับปลั๊กและสายเคเบิลที่ใช้กับเครื่องไมโครคอมพิวเตอร์นั้นอาจเป็นข้อยกเว้น คือเราใช้มาตรฐานเดียวกันเกือบทุกยี่ห้อ เราจึงพอจะเดาสุ่ม ๆ แล้วเชื่อมโยงกันได้โดยไม่ต้องดูยี่ห้อ แต่ท่านอย่าไปเหมาเอาว่าเครื่องเมนเฟรม 2 ยี่ห้อจะต่อกันได้ง่าย ๆ ปัญหาที่ต้องระวังในขั้นสุดท้ายก็คือ ระดับแรงดันของสัญญาณจากเครื่องทั้งสองจะต้องตรงกันด้วย ที่สร้างปัญหาได้มาก ๆ ก็คือ แรงดันเหล่านี้ถูกดูดหรือหน่วงให้ลดลงโดยสายเคเบิล ซึ่งจะมีผลทำให้แม้แต่ฮาร์ดแวร์ยี่ห้อเดียวกันก็อาจพูดกันไม่รู้เรื่องได้

การเชื่อมโยงไม่ใช่เรื่องง่าย

โอเค สมมติว่าท่านหาช่างมาศึกษาจนสามารถเชื่อมโยงในระดับฮาร์ดแวร์ได้แล้ว แล้วท่านจะทำอย่างไรต่อไป? เครื่องคอมพิวเตอร์ 2 เครื่องที่เชื่อมโยงทางสายสื่อสารได้แล้วนั้นก็เหมือนกับเอาคน 2 คนมายืนอยู่คนละฝั่งคลอง แต่ทั้ง 2 ยังไม่มีปากและไม่มีหู ที่เป็นเช่นนี้ก็เพราะเรายังไม่มีอะไรที่จะไปทำให้เกิดสัญญาณบนสายสื่อสารได้เลย เครื่องคอมพิวเตอร์ต้องมีโปรแกรมวิ่งและโปรแกรมนั้นจะต้องมีความสามารถในการรับสัญญาณจากสายสื่อสาร นี่คือสิ่งที่ผู้บริหารและเซลส์แมนหรือเซลส์วูแม่นลืมนึกถึง เซลส์แมนและเซลส์วูแม่นอาจรู้ว่าต้องมีโปรแกรม แต่เขาไม่รู้หรอกว่าจะทำให้โปรแกรมนั้นพูดและฟังได้ถูกต้องนั้นยาก

โปรแกรมที่เรามีอยู่ หรือที่โปรแกรมเมอร์ทั่ว ๆ ไปรู้จักเขียนนั้นมี 2 แบบ แบบแรกเรียกว่า "งานแบตช์" (Batch Processing) โปรแกรมประเภทนี้รู้จักพูดกับจานแม่เหล็ก เทป และเครื่องพิมพ์ หรือคือ รู้จักพูดลงไปในกระบอกไม้เท่านั้น คนที่อยู่คนละฝั่งคลองจะไม่มีทางได้ยิน โปรแกรมประเภทที่ 2 ที่พวกเรารู้จักเขียนกันเรียกว่า "งานออนไลน์" (On-line Processing) โปรแกรมประเภทนี้เก่งกว่าประเภทแรกอีกนิดหน่อย คือสามารถพูดกับเทอร์มินัลโดยผ่านสายสื่อสารได้ด้วย แต่มันเป็นการส่งสัญญาณแบบ "นายกับบ่าว" เท่านั้น กล่าวคือ เราจะเริ่มต้นด้วยการใส่โปรแกรมด้านการควบคุมการสื่อสารลงไปก่อน เสร็จแล้วเจ้าโปรแกรมนี้จะสร้างสัญญาณขึ้นไปบนสายสื่อสารว่ามีอะไรจะส่งมาจากเทอร์มินัลหรือไม่ เทอร์มินัลที่เปิดทิ้งไว้เฉย ๆ ก็จะตอบว่า "ไม่มี" สิ่งนี้จะวนเวียนไปเรื่อย ๆ จะวนจนกว่าจะมีใครสักคนหนึ่งไปกดแป้นคีย์บอร์ดที่เทอร์มินัลและกดแป้น "ส่งข้อมูล" (Transmit) ถึงคราวนี้เจ้าเทอร์มินัลจะตอบว่า "มี" เมื่อถูกถามมาอีก แล้วเจ้าโปรแกรมที่ควบคุมระบบสื่อสารก็จะจัดการเตรียมรับข้อมูลและส่งสัญญาณไปบอกให้เทอร์มินัลนั้นเริ่มส่งข้อมูลได้ เมื่อเทอร์มินัลได้รับอนุญาตแล้วก็จะส่งสัญญาณเป็นตัวอักษรตามที่เรากดแป้นคีย์บอร์ดเข้าสู่โปรแกรมที่ควบคุมการสื่อสาร โปรแกรมควบคุมการสื่อสารได้รับแล้วก็จะส่งข้อความนั้นต่อไปยังโอเปอเรติ้งซิสเต็มส์ หรือไปยังโปรแกรมของเรา ฯลฯ การติดต่อแบบนี้

CONNECTIVITY ผูกใจคอมพิวเตอร์เป็นหนึ่งเดียว

เหมือนกับการพูดเป็นภาษาพิเศษ คือมีการตกลงกันไว้ล่วงหน้าว่า ถ้าฉันออกเสียงว่า "อือ-อือ" ก็ให้หมายความว่า "มีอะไรจะส่งเข้ามาไหม" และเมื่อเทอร์มินัลออกเสียงว่า "อี-อี" ก็ให้หมายความว่า "ยังไม่มี" แต่ถ้าตอบว่า "ออ-ออ" ก็ให้หมายความว่า "มี" ฯลฯ และท่านควรสังเกตว่าโปรแกรมควบคุมการสื่อสารนั้นทำตัวเป็น "นาย" ส่วนเทอร์มินัลนั้นเป็น "บ่าว" ถ้านายไม่ถามมาบ่าวก็จะพูดไม่ได้เลย การพูดกันแบบนี้ใช้กับคน 2 คนที่อยู่คนละฝั่งคลองไม่ได้ ทั้งนี้เพราะถ้าเป็นเครื่องคนละยี่ห้อต่างคนต่างก็จะเป็นนาย และนายคนละยี่ห้อนั้นเขาใช้รหัสเสียงไม่เหมือนกัน ถึงแม้ว่าหลักการและขั้นตอนจะคล้ายกัน แต่ก็พูดกันไม่รู้เรื่อง

การเชื่อมโยงระดับซอฟต์แวร์

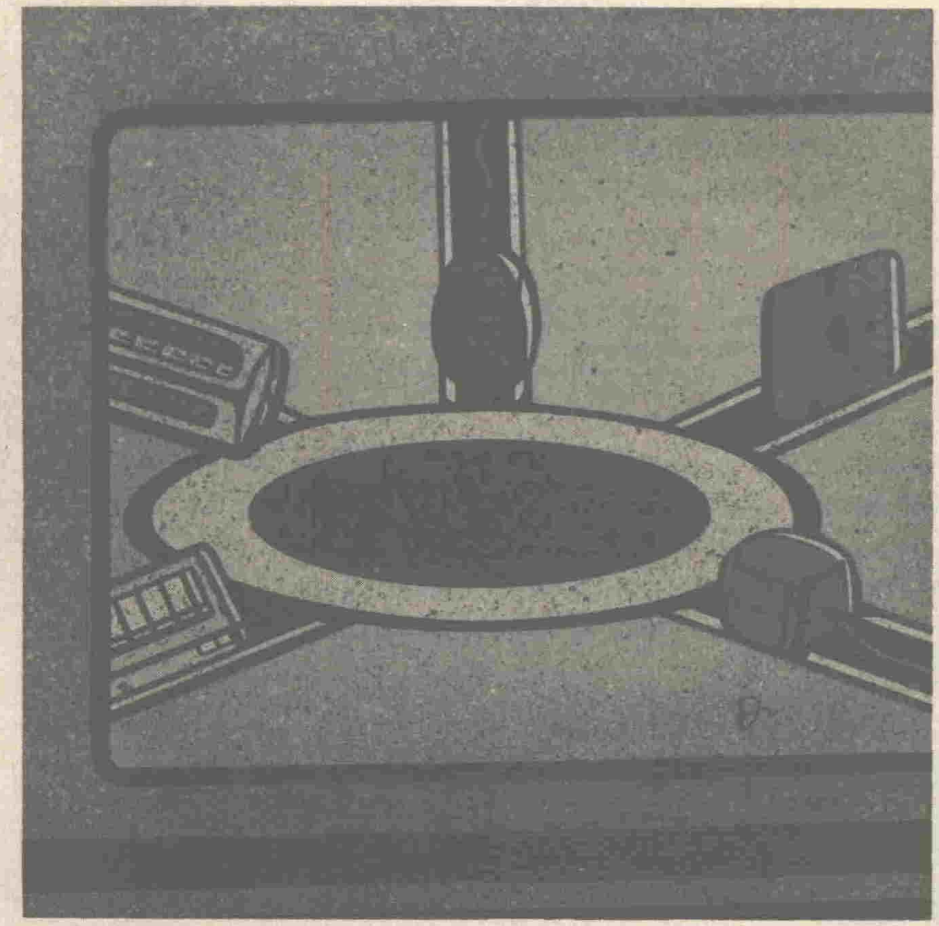
โดยสรุปแล้วโปรแกรมที่เราโหลดลงไปบนซีพียูและในตัวควบคุมการสื่อสารนั้น เปรียบเสมือนเป็นปากและหู แต่โปรแกรมที่พวกเรารู้จักเขียนขึ้นมาใช้งานกันนั้นเป็นการพูดใส่กระบอกหรือไม่ก็ใช้ขั้นตอนและรหัสเสียงที่ไม่เหมือนกัน เราจึงไม่สามารถจะทำให้เครื่อง 2 เครื่องนั้นส่งถ่ายข้อมูลกันได้

แล้วเราจะทำอย่างไรกันต่อไป?

ก็ต้องเขียนโปรแกรมใส่เข้าไปในซีพียูทั้งสอง และต้องพูดภาษาเดียวกันสิครับ นี่คือการเชื่อมโยงในระดับซอฟต์แวร์ การพูดภาษาเดียวกัน และมีขั้นตอนการรับส่งข้อมูลที่ผสมผสานกันได้นี้เราเริ่มต้นโดยทำกับเครื่องยี่ห้อเดียวกันก่อน เรื่องนี้ทำกันมาเกือบ 20 ปีแล้ว เรามีความสำเร็จในต่างประเทศอยู่พอสมควร แต่ในเมืองไทยนั้นเกือบไม่มีใครทำได้เลย เรามีคนทำกับเครื่องยี่ห้อไอบีเอ็มระดับเมนเฟรมอยู่ 2-3 แห่ง ความรู้ในเรื่องนี้มีจำกัดมาก สำหรับการเชื่อมโยงระดับซอฟต์แวร์ของเครื่องคนละยี่ห้อนั้นฝรั่งก็ได้พยายามทำกัน เรื่องนี้ต้องมีการประชุมหารือร่วมกันหลาย ๆ ยี่ห้อ ผู้ที่ช่วยเป็นแกนนำในการกำหนดมาตรฐานเหล่านี้ได้แก่ "ไอทริปเปิ้ลอี" (IEEE) และ "ไอ-เอสโอ" (ISO) การกำหนดมาตรฐานเหล่านี้เป็นไปได้ด้วยดี แต่เรามีปัญหาในแง่ของการสร้างซิสเต็มส์ซอฟต์แวร์ที่ตรงกับข้อกำหนด ที่เป็นเช่นนี้ก็เพราะข้อกำหนดเหล่านี้ค่อนข้างซับซ้อน และไม่มีหนทางที่จะทำให้มันง่ายกว่านี้ได้ ผมจะขอยกตัวอย่างให้ฟังสัก 1-2 เรื่อง

ตัวอย่างประกอบ

ในการที่จะให้โปรแกรมทำงาน (Application) ของเราในเครื่องที่ 1 สามารถติดต่อกับโปรแกรมทำงาน (Application) ของเราในเครื่องที่ 2 ได้นั้น เราต้องทำให้โปรแกรมทั้งสองรู้จักกันเสียก่อน เรื่องนี้เราทำได้โดยให้โปรแกรมข้างใดข้างหนึ่งส่งคำร้องขอผ่านมายังโอเปอเรติงซิสเต็มส์ของอีกเครื่องหนึ่ง เราเรียกข้อความอันนี้ว่า "เฮดเดอร์" (Header) โอเปอเรติงซิสเต็มส์จะเช็คดูว่ามีโปรแกรมชื่อตามที่ขอเชื่อมโยงอยู่ในเมมโมรีของตนหรือไม่ เมื่อพบว่ามีก็จะส่งต่อเฮดเดอร์ดังกล่าวไปให้ โปรแกรมที่ได้รับการร้องขอก็จะต้องจดบันทึกไว้ว่ามีโปรแกรมชื่อ XX... ได้ติดต่อมา และจะต้องจัดเตรียมเพื่อการโต้ตอบกันต่อไป เสร็จ



แล้วก็ต้องแจ้งผลกลับไป เรื่องนี้จะถูกส่งผ่านกลับไปยังโปรแกรมผู้ร้องขอ โปรแกรมผู้ร้องขอก็จะจัดเนื้อที่เพื่อรองรับการโต้ตอบกันต่อไป ขั้นตอนทั้งหมดที่กล่าวมานี้เราเรียกกันง่าย ๆ ว่า "การจับมือกัน" (Hand shaking) เมื่อจับมือกันได้แล้วจึงจะสามารถส่งข้อมูลให้แก่กันและกันได้

โอเค หลังจากจับมือกันไว้อย่างมั่นเหมาะแล้วเราจะทำอะไรกันได้บ้าง? สิ่งที่เราทำกันบ่อยมากก็คือ ขอให้โปรแกรมในเครื่องที่ 1 อ่านข้อมูลจากแฟ้มข้อมูลที่อยู่ในจานแม่เหล็กของเครื่องที่ 1 แล้วส่งไปให้โปรแกรมที่ 2 ที่อยู่ในเครื่องที่ 2 และให้โปรแกรมที่ 2 นั้นบันทึกข้อมูลดังกล่าวลงไปบนจานแม่เหล็กของเครื่องที่ 2 งานชนิดนี้เราเรียกว่า "การโยกย้ายข้อมูล" (File Transfering) ในการทำงานดังกล่าวนี้เราไม่สามารถที่จะให้โปรแกรมที่ 1 อ่านข้อมูลขึ้นมาทีเดียวจนหมดแฟ้มข้อมูลทั้งนี้เพราะแฟ้มข้อมูลส่วนใหญ่มีขนาด (จำนวนข้อมูล) ใหญ่เกินกว่าที่จะเอาขึ้นมาบรรจุลงในเมมโมรีของซีพียูได้หมด เราต้องอ่านทีละเรคอร์ด คำว่าเรคอร์ดนี้ก็เปรียบได้เท่ากับข้อความ 1 บรรทัด เมื่อได้บรรทัดแรกแล้วก็จัดการส่งไปให้โปรแกรมที่ 2 เสร็จแล้วก็จะวนกลับไปอ่านขึ้นมาอีก 1 บรรทัด แล้วก็จัดการส่งไปให้โปรแกรมที่ 2 อีก ฯลฯ นี่คือขั้นตอนการทำงานของโปรแกรมที่ 1 ในขณะเดียวกันเราก็ต้องเขียนโปรแกรมที่ 2 ให้ทำงานเป็นขั้นตอนดังนี้คือ ให้รับข้อมูลจากโปรแกรมที่ 1 ที่ผ่านมาทางสายสื่อสาร รับได้ 1 บรรทัดแล้วก็จัดการบันทึกลงไปที่จานแม่เหล็ก เสร็จแล้วก็หมุนวนกลับไปรับข้อมูลบรรทัดที่ 2 ใหม่อีก โปรแกรมนี้จะหมุนวนอย่างนี้ไปเรื่อย ๆ จนกว่าจะหมดข้อมูล

ปัญหาที่ไม่ควรมองข้าม

ครับ ฟังดูก็ง่ายดี แต่ขอให้ท่านสังเกตว่าข้อมูลแต่ละบรรทัดที่โปรแกรมที่ 1 อ่านขึ้นมานั้นจะทับลงไปบนเนื้อที่เดิมในเมมโมรีของเครื่องที่ 1 และเช่นเดียวกัน โปรแกรมที่ 2 นั้นเวลารับข้อมูลเข้ามาก็อาจต้องทับลงไปบนเนื้อที่เดิมในเมมโมรีของเครื่องที่ 2 ที่เป็นเช่นนี้ก็เพราะเราไม่มีเมมโมรีเพียงพอที่จะรองรับข้อมูลทุกบรรทัดได้ แต่ขณะเดียวกัน ท่านก็ควรสังเกตว่าโปรแกรมที่ 1 กับโปรแกรมที่ 2 นี้อยู่บนเครื่องคอมพิวเตอร์คนละเครื่องและอาจเป็นคนละยี่ห้อ ดังนั้นการทำงานจึงอาจช้าเร็วต่างกันได้ เรามาลองสมมติว่า เครื่องที่ 1 ทำงานได้เร็วกว่าเครื่องที่ 2 ดูก่อน ถ้าเป็นกรณีนี้ก็จะพบว่า ข้อมูลบรรทัดที่ 2 ที่ส่งมายังโปรแกรมที่ 2 นั้นจะมาทับบางส่วนของบรรทัดที่ 1 ทั้งนี้เพราะโปรแกรมที่ 2 ยังไม่ทันเอาข้อมูลทั้งหมดของบรรทัดที่ 1 ไปเก็บลงในจานแม่เหล็ก การทำงานภายในเมมโมรีนี้มันทำงานทีละ 2 ตัวอักษร (2 ไบต์) และสามารถสลับเวลาให้จานแม่เหล็กกับระบบสื่อสารทำงานขนานกันได้ เหตุการณ์เช่นนี้จะทำให้แฟ้มข้อมูลที่ได้รับในเครื่องที่ 2 นั้นไม่ตรงกับในเครื่องที่ 1 มันจึงไม่เกิดประโยชน์อันใด ในทางตรงกันข้าม ถ้าเราสมมติว่าเครื่องที่ 2 นั้นทำงานเร็วกว่าเครื่องที่ 1 ก็จะพบว่าเครื่องที่ 2 อาจวนกลับมาอ่านเอาข้อมูล

บรรทัดที่ 1 บันทึกลงเป็นบรรทัดที่ 2 ในจานแม่เหล็กด้วย ซึ่งก็เสียหายเช่นกัน

แนวทางแก้ไขปัญหา

โดยสรุปแล้ว แนวคิดแบบง่าย ๆ อย่างที่ผมสมมติให้ฟังในตอนต้นนั้นใช้การไม่ได้ เรื่องนี้ต้องแก้ไขโดยให้มีซอฟต์แวร์อีกตัวหนึ่งขึ้นมาแทรกระหว่างโปรแกรมของเรากับโปรแกรมควบคุมการสื่อสาร ซอฟต์แวร์ตัวนี้จะเป็นตัวรองรับข้อมูลที่จะรับและส่งกัน ซอฟต์แวร์ตัวนี้มีหน้าที่จัดคิวของข้อมูล เช่น เมื่อรับข้อมูลจากโปรแกรมที่ 1 มาได้ 1 บรรทัด มันก็จะเอาไปเก็บไว้ที่ไหนสักแห่งหนึ่งก่อน เสร็จแล้วทำคำสั่งส่งไปให้ตัวควบคุมการสื่อสารให้ส่งออกไปยังเครื่องที่ 2 ทำคำสั่งแล้วก็จะเวียนมารับข้อมูลจากโปรแกรมที่ 1 ใหม่อีก ในโปรแกรมที่ 1 ของเรานั้นก็ต้องปรับแก้ คือก่อนที่จะวนกลับมาอ่านบรรทัดที่ 2 ก็ต้องเช็คว่าการสั่งบรรทัดที่ 1 นั้นได้รับการจัดการ (OK Signal) จากซอฟต์แวร์ที่กล่าวถึงแล้วหรือไม่ ถ้ายังไม่เรียบร้อยก็ต้องรอสักเล็กน้อยแล้วสอบถามใหม่ เมื่อพบว่าการส่งบรรทัดที่ 1 ได้รับการดำเนินการแล้วจึงจะหมุนวนไปอ่านบรรทัดที่ 2 ขึ้นมา แล้วจัดส่งออกไปอีก เจ้าซอฟต์แวร์ตัวจัดคิวนั้นจะไปควานหาเนื้อที่ในเมมโมรีเพื่อรองรับข้อมูลบรรทัดที่ 2 ให้ มันอาจไปใช้เนื้อที่เดียวกันกับบรรทัดที่ 1 ก็ได้ถ้าบรรทัดที่ 1 ถูกส่งออกไปหมดแล้ว แต่ถ้ายังไปไม่หมด (เพราะเครื่องที่ 2 ทำงานช้ากว่า) มันก็จะไปจัดหาเนื้อที่ใหม่ในเมมโมรีมาใช้งาน

นั่นเป็นโปรแกรมจัดคิวทางด้านเครื่องที่ 1 สำหรับเครื่องที่ 2 นั้นก็จะมีโปรแกรมจัดคิวเช่นกัน แต่มันจะทำงานตรงกันข้ามกับในเครื่องที่ 1 คือ เมื่อได้รับข้อมูลบรรทัดที่ 1 เข้ามาก็จะนำไปเก็บไว้ที่ไหนสักแห่งหนึ่ง เสร็จแล้วจดบันทึกไว้ว่าเป็นของโปรแกรมที่ 2 เมื่อใดที่โปรแกรมที่ 2 มาขอรับข้อมูลก็จะจัดส่งไปให้ ในขณะที่รอคอยโปรแกรมที่ 2 อยู่นั้น โปรแกรมจัดคิวนี้ก็อาจได้รับข้อมูลบรรทัดที่ 2 เข้ามา มันก็จะไปหาเนื้อที่ในเมมโมรีเพื่อเก็บเอาไว้อีก มันอาจใช้เนื้อที่ที่เคยเก็บข้อมูลบรรทัดที่ 1 ก็ได้ถ้าข้อมูลบรรทัดที่ 1 ถูกโปรแกรมที่ 2 เรียกไปใช้แล้ว แต่ถ้าข้อมูลบรรทัดที่ 1 ยังไม่ถูกเรียกไปใช้มันก็ต้องไปหาเนื้อที่ใหม่ เมื่อรับไว้แล้วก็จดบันทึกไว้ว่าเป็นของโปรแกรมที่ 2 และเป็นตัวอยู่ในคิวเป็นอันดับ 2 คิวนี้จะเป็นระบบ "มาก่อน-ออกก่อน" (First-In-First-Out) โปรแกรมจัดคิวทั้งสองด้านนี้จริง ๆ แล้วเขาจะจับมารวมกัน ทั้งนี้เพื่อให้เราสามารถทำงานย้อนกลับได้ คือให้เครื่องที่ 2 เป็นตัวส่งข้อมูล และเครื่องที่ 1 เป็นตัวรับข้อมูล ยิ่งไปกว่านั้นโปรแกรมจัดคิวนี้จะต้องขยายไปรับภาระให้กับทุกโปรแกรมได้ด้วย โปรแกรมแบบนี้เราจัดเป็นส่วนหนึ่งของ "โปรแกรมระบบควบคุม" (System Software)

ความซับซ้อนของซอฟต์แวร์

เป็นยังไงครับ ท่านจะเห็นว่า แค่ที่เล่ามาให้ฟังนี้ก็ซับซ้อนพอสมควรแล้ว แต่ท่านอาจไม่ทราบว่านี้เป็นเพียง 1 ใน 100 ของความซับซ้อนที่มีอยู่ ทั้งนี้เพราะเรายังไม่ได้พูดถึงว่า โปรแกรมนี้จะรู้ได้อย่างไรว่าข้อมูลที่รับเข้ามานั้นเป็นของโปรแกรมไหน และถ้าโปรแกรมใดโปรแกรมหนึ่งเกิดหยุดไปเสียก่อนที่จะวนมารับข้อมูลไปจนหมดจะทำอย่างไร ข้อมูลที่ค้างอยู่ในคิวเหล่านี้จะปล่อยให้อยู่ไปจนกาลอวสานหรืออย่างไร และที่สำคัญมาก ๆ ก็คือ ถ้าข้อมูลมาค้างอยู่ในคิวแยะ ๆ จนไม่มีเมมโมรีที่จะรับเข้ามาได้อีกเราจะทำอย่างไรกันต่อไป

ครับ นั่นคือความซับซ้อนของซอฟต์แวร์ที่จะทำให้เกิดการรับส่งข้อมูลระหว่างเครื่อง 2 เครื่อง และความซับซ้อนอันนี้จะมีผลกระทบต่อการออกแบบขั้นตอนการทำงานของโปรแกรมงาน (Application) ของเราด้วย เราต้องมาเรียนรู้คำสั่งใหม่ ๆ อีกจำนวนหนึ่ง และเราต้องเรียนรู้เทคนิคใหม่ ๆ อีกจำนวนหนึ่ง (เช่นอย่าจบโปรแกรมโดยยังไม่ได้เช็คว่ามีข้อมูลค้างอยู่ในระบบสื่อสารหรือไม่ และอาจต้องเช็คข้ามไปจนถึงโปรแกรมในอีกเครื่องหนึ่งว่าได้จัดการกับข้อมูลที่ส่งไปจนเรียบร้อยหรือไม่ ฯลฯ) สิ่งเหล่านี้ไม่เกินขีดความสามารถของโปรแกรมเมอร์ที่เรามีอยู่ แต่ท่านก็ต้องให้เวลากับพวกเขา เขาอาจต้องใช้เวลา 6 ถึง 24 เดือนเพื่อทำความคุ้นเคยกับซิสเต็มส์ซอฟต์แวร์ใหม่ ๆ เหล่านี้ และเขาอาจต้องใช้เวลาในการออกแบบและพัฒนาโปรแกรมเพื่อให้ท่านใช้งานอีก 10 ถึง 20 เดือน

โครงการเชื่อมโยงคอมพิวเตอร์ (ให้ถึงระดับ Application) นี้ต้องใช้เวลา 3 ถึง 5 ปี และไม่มีใครทำได้ดีเท่ากับให้คนทางด้านคอมพิวเตอร์ของท่านทำ การไปจ้างผู้เชี่ยวชาญจากต่างประเทศเข้ามาพัฒนาให้นั้นจะใช้เวลาไม่ยิ่งหย่อนกว่ากัน แต่มีค่าใช้จ่ายกว่า 10 เท่า โครงการ "ระบบชำระเงิน" (Payment System) ของธนาคารแห่งประเทศไทยจะเป็นตัวอย่างให้ดูได้ ในโครงการนี้จะมีเครื่องเมนเฟรมและเครื่องประเภท "รับส่งข้อมูล" (Message Switch) อยู่ 3-4 ตัว ซึ่งจะต้องทำงาน

เชื่อมโยงกัน และจะมีขีดความสามารถให้ธนาคารพาณิชย์ต่าง ๆ เอาเมนเฟรมที่มีอยู่มาเชื่อมโยงเข้าไปได้ โครงการนี้ใช้เวลาศึกษาและเปิดประมูล 1 ปี บริษัทผู้ได้รับว่าจ้างจะต้องใช้เวลาอีก 21 เดือน เพื่อพัฒนาโปรแกรม แล้วยังต้องรอให้ธนาคารพาณิชย์ต่าง ๆ พัฒนาฮาร์ดแวร์และซอฟต์แวร์เพื่อเชื่อมโยงเข้าไปอีก X เดือน ตัวเลข X นี้จะเป็นเท่าใดเรายังไม่รู้ แต่ระบบที่คล้าย ๆ กันนี้มีในญี่ปุ่น เขาใช้เวลาพัฒนากัน 5 ปี

ทางออกของ Connectivity

นั่นคือ "Connectivity" ในระดับเมนเฟรมและมินิคอมพิวเตอร์ จากภาพที่ผมเล่ามาให้ฟังนี้อาจทำให้หลาย ๆ คนท้อใจ แต่เรายังมีทางออกที่ดี ๆ อีกหลายทาง ผมจะขอถือโอกาสนี้เอามาอธิบายอย่างย่อ ๆ

ทางออกที่ 1

ทางออกทางแรกนั้นมาจากกลุ่ม "ยูนิกซ์" (UNIX) กลุ่มนี้ดั้งเดิมมาจากทีมอาจารย์ในมหาวิทยาลัยต่าง ๆ พวกเขาสนใจที่จะเชื่อมโยงเครื่องคอมพิวเตอร์หลาย ๆ ยี่ห้อและหลาย ๆ ระดับเข้าหากัน เหตุผลใหญ่ก็เป็นเพราะมหาวิทยาลัยต่าง ๆ ได้รับบริจาคเครื่องมาหลายยี่ห้อและหลายระดับ เขามีทั้งเมนเฟรม, มินิฯ, ไมโครฯ และไฮเพาเวอร์เวอร์กสเตชัน และอาจารย์เหล่านี้ต้องการแลกเปลี่ยนข้อมูลและข่าวสารซึ่งกันและกัน แนวคิดของคนกลุ่มนี้ก็คือ "มาช่วยกันสร้างระบบควบคุม (Operating System) ที่สามารถใช้คำสั่ง (Command) ที่เป็นมาตรฐานเดียวกัน" ด้วยวิธีนี้เราสามารถโยกย้ายงานจากเครื่องยี่ห้อหนึ่งไปยังเครื่องอีกยี่ห้อหนึ่งได้โดยง่าย เราเรียกเป็น "ระบบเปิด" (Open System)

นอกจากนี้แล้ว เงื่อนไขที่สำคัญมากประการหนึ่งก็คือ "ระบบควบคุมอันนี้จะต้องเปิดโอกาสให้มีการแลกเปลี่ยนข้อมูลกันแบบออนไลน์ได้โดยสะดวก" ระบบนี้เราเรียกว่า "ยูนิกซ์" หรือ "ยูนิกซ์โอเปอเรติงซิสเต็มส์" ทีมดังกล่าวนี้ได้ก่อตั้งขึ้นมากว่า 20 ปีแล้ว และได้รับการสนับสนุนจากกลุ่มผู้ผลิตเครื่องมือทางการสื่อสารด้วย เช่น บริษัท เอทีแอนด์ที แต่จริง ๆ แล้วเราจะพบว่าทีมงานดังกล่าวไม่สามารถไปพัฒนาโปรแกรมที่เป็นโอเปอเรติงซิสเต็มส์ได้ ทั้งนี้เพราะเครื่องแต่ละรุ่นแต่ละยี่ห้อจะเป็นโปรแกรมที่ใช้คำสั่งระดับล่าง (Machine Code) ที่ต่าง ๆ กัน พวกเขาเพียงแต่ร่างข้อกำหนดว่าโปรแกรมโอเปอเรติงซิสเต็มส์เหล่านี้จะทำอะไรได้บ้าง เช่น มีคำสั่งที่เราจะกดแป้นสั่งเข้าไปอย่างไรได้บ้าง ขั้นตอนการโต้ตอบกันระหว่างโปรแกรมบนเครื่อง 2 เครื่องที่โยงถึงกันนั้นจะเป็นอย่างไร ระบบป้องกันข้อมูลและการกู้ระบบจะเป็นอย่างไร ฯลฯ เสร็จแล้วส่งให้บริษัทต่าง ๆ ไปพัฒนาเข้าใส่ในเครื่องยี่ห้อของตน

ระบบยูนิกซ์นี้ทำให้เกิดการเชื่อมโยงได้ง่ายขึ้น แต่อย่างไรก็ตาม ในขณะนี้บริษัทต่าง ๆ ยังไม่สามารถทำได้ตรงกับข้อกำหนดถึง 100 เปอร์เซ็นต์ ผู้ที่ซื้อเครื่อง 2 ยี่ห้อมาเชื่อมโยงกันโดยใช้ยูนิกซ์เป็นตัวควบคุมนี้จึงยังมีความเสี่ยงที่จะไม่สำเร็จอยู่บ้าง และระบบยูนิกซ์นี้ก็ไม่ใช่ระบบที่จะเรียนรู้ได้ง่ายนัก เราอาจต้องใช้เวลา 1-2 ปี เพื่อ

พัฒนาโปรแกรมงาน (Application) นอกจากนี้แล้วท่านก็อย่าเพิ่งด่วนสรุปว่าการเชื่อมโยงเครื่องยี่ห้อเดียวกันจะทำได้โดยราบรื่น ท่านอาจต้องผจญกับ "บัก" ในซิสเต็มส์ซอฟต์แวร์เหล่านี้ ซึ่งจะทำให้โครงการของท่านต้องล่าช้าไปได้อีกนับเป็นปี ๆ มองภาพในขณะนี้จึงไม่ดีกว่าวิธีแรก (ใช้เมนเฟรมและมินิและใช้ซิสเต็มส์ซอฟต์แวร์แบบ Proprietory) เท่าใดนัก แต่มีแนวโน้มว่าจะดีขึ้นในอนาคต อนาคตนี้จะใกล้หรือไกลก็ขึ้นอยู่กับความร่วมมือของผู้ผลิตต่าง ๆ ซึ่งในขณะนี้ดูจะไม่ดีนัก

ทางออกที่ 2

ทางออกอันที่ 2 คือรอให้บริษัทต่าง ๆ พัฒนาซิสเต็มส์ซอฟต์แวร์ด้านการสื่อสารตามมาตรฐานโลก มาตรฐานที่เป็นที่ยอมรับกัน ได้แก่ LU 6.2, X.25, IEEE 802.3 มาตรฐานเหล่านี้เป็นกฎเกณฑ์ในด้านการรับส่งข้อมูลกันหรือคือเน้นทางด้านการสื่อสารข้อมูลคอมพิวเตอร์ แต่ตัวโปรแกรมงาน (Application) จะวิ่งอยู่ภายใต้การควบคุมของโอเปอเรติงซิสเต็มส์ของแต่ละยี่ห้อ ข้อนี้ทำให้เราไม่สามารถโยกย้ายงานข้ามยี่ห้อได้ แต่ก็สามารถให้เราเชื่อมโยงเครื่องต่างยี่ห้อกันได้ ซิสเต็มส์ซอฟต์แวร์ประเภทนี้ถูกผลิตออกมาขายกันมากขึ้น มันสามารถนำมาใช้แทนโปรแกรมควบคุมการสื่อสารแบบเก่า ๆ (SDLC, HDLC) ได้ และมีความเชื่อถือได้สูงขึ้น ในระบบควบคุมการสื่อสารแบบเก่า ๆ นั้น เมื่อมีการรับส่งข้อมูลกันมาก ๆ มักจะตายได้ สาเหตุนั้นมักจะมาจากการจัดการกับเนื้อที่ในเมมโมรีผิดพลาด (ไม่มีเนื้อที่พอที่จะรองรับข้อมูลไว้ในคิว) ซอฟต์แวร์รุ่นใหม่จะจัดการกับเรื่องเหล่านี้ได้ดีกว่า แต่ก็มีราคาแพงขึ้นอีกเล็กน้อย ซอฟต์แวร์ประเภทนี้ทำให้เกิด "Connectivity" ข้ามยี่ห้อได้ ข้อดีของหนทางนี้คือ เมื่อเราพัฒนาจนเข้าที่ดีแล้วก็จะไม่ต้องแก้ไขอีกเลย มันจะทำงานเหมือนกับเป็นส่วนหนึ่งของฮาร์ดแวร์ (อาจเรียกว่า Firmware) ตัวปลั๊กและสายสัญญาณต่าง ๆ ก็จะเป็นตามมาตรฐาน เราสามารถซื้อจากคนละยี่ห้อแล้วมาเสียบต่อกันได้เลย เราจะต้องไม่ใช้ช่างระดับพระกาฬเหมือนในวิธีแรก

ทางออกที่ 3

ทางออกอันที่ 3 คือซื้อเทคโนโลยีด้านสื่อสารจากบริษัทผู้ผลิตสินค้าและบริการการสื่อสาร ทางออกทางนี้เรามักจะมองข้ามกันอยู่เสมอ ๆ บริษัทผู้ผลิตระบบสื่อสารนั้นเริ่มจากการให้บริการเชื่อมโยงสายสื่อสาร แต่บริการของเขามักจะพบกับปัญหาการสอดแทรกของสัญญาณ และอาจเจอกับปัญหาสายสื่อสารขาดไปเป็นครั้งคราว การรบกวนต่าง ๆ เหล่านี้ไปสร้างปัญหาให้กับโปรแกรมงาน (Application) เช่น เพื่อให้แน่ใจว่าข้อมูลที่ได้รับไปนั้นไม่มีผิดเพี้ยนเราก็ต้องใส่รหัสข้อมูลเพิ่มเติมเพื่อเอาไว้สอบยัน (Error Detection and Error Correction Codes) แล้วโปรแกรมของเราก็ต้องตรวจเช็คหรือเพื่อให้แน่ใจว่าข้อมูลไม่ได้ขาดหายไปทั้งบรรทัด เราก็ต้องเพิ่มการนับจำนวนและตอบยืนยันกัน และเพื่อป้องกันปัญหาจากการขาดของสายสื่อสาร เราก็ต้องมีการจับเวลาและหาทางส่งข้อมูลไปทางอื่น ปัญหาต่าง ๆ เหล่านี้แก้ไขได้

โดยใส่ฮาร์ดแวร์เข้าไปที่จุดเชื่อมต่อกับคอมพิวเตอร์ทุกแห่ง ฮาร์ดแวร์เหล่านี้ก็เป็นคอมพิวเตอร์ชนิดหนึ่ง เราเรียกมันว่า "ตัวควบคุมจุดเชื่อมโยง" (Node หรือ Node Controller) นอกจากนี้เขาก็ยังเพิ่มสายสื่อสารระหว่างตัวควบคุมจุดเชื่อมโยงเหล่านี้เข้าไปด้วย สายสื่อสารเพิ่มเติมเหล่านี้มีไว้เพื่อเป็นทางผ่านฉุกเฉิน ซอฟต์แวร์ที่วิ่งอยู่บนตัวควบคุมจุดเชื่อมโยงเหล่านี้ จะทำหน้าที่แก้ปัญหาให้เราจนหมด เขาสามารถรับประกันว่าข้อมูลทุกบรรทัดจะถูกส่งไปให้ถึงที่หมายได้แน่นอน และได้ถูกต้องและครบถ้วนทุกประการ แต่อาจถูกเก็บพักไว้เป็นระยะเวลาหนึ่งเมื่อสายสื่อสารบางช่วงตอนเสียหาย

เทคโนโลยีด้านนี้กำลังได้รับการพัฒนาอย่างรวดเร็ว บริษัทเหล่านี้ได้รับจ้างทำระบบสื่อสารขนาดใหญ่ให้กับลูกค้ารายใหญ่ ๆ เสียก่อน เมื่อมีความรู้ดีแล้วก็พยายามจะผลิตออกขายเป็นสินค้ามาตรฐาน ลูกค้าทั่วไปไปหาซื้อมาใช้กันได้ และพวกนี้เขาผลิตตามมาตรฐานที่เขาร่วมมือกันจัดขึ้น อุปกรณ์ของยี่ห้อต่าง ๆ จึงนำมาใช้ร่วมกันได้พอสมควร ทางออกทางนี้ดูจะแจ่มใสขึ้นเรื่อย ๆ ถึงแม้ว่าการลงทุนครั้งแรกดูจะแพงมาก (Node, Bridge, Router ฯลฯ เหล่านี้เป็นเครื่องคอมพิวเตอร์ขนาดเล็กและขนาดกลาง ราคาแต่ละชิ้นหลายแสนบาท) แต่มองดูในระยะยาว เช่น การพัฒนาโปรแกรมทำได้ง่ายขึ้น การเชื่อมโยงเครื่องอื่น ๆ เข้ามาในเครือข่ายจะง่ายขึ้น เราก็จะพบว่ามันเป็นวิธีที่ดีมากทีเดียว โปรแกรมจะสั้นขึ้น โครงสร้างของโปรแกรมจะย้อนกลับไปสู่สภาพแบบง่าย ๆ การทดสอบระบบจะทำได้ง่ายและเร็วขึ้น เรื่องนี้ผู้เขียนยังไม่มีประสบการณ์ จึงบอกไม่ได้ว่าจะทำให้เราพัฒนาโปรแกรมได้เร็วขึ้นเป็นเท่าใด แต่เรากำลังจะมีตัวอย่างให้ดูที่จุฬาลงกรณ์มหาวิทยาลัย ชุดนี้เป็นแบบ "FDDI" หรือคือใช้เส้นใยแสงเป็นตัวสายสื่อสาร ภายในเครือข่ายนี้จะมีตัวควบคุมและสับเปลี่ยนทิศทางข้อมูลวางอยู่ตามคณะต่าง ๆ ระบบนี้จะเปิดโอกาสให้อาจารย์ในคณะต่าง ๆ ใช้เครื่องคอมพิวเตอร์เชื่อมโยงถึงกันได้หมด ความเร็วในการรับส่งกันจะอยู่ที่ 10 ล้านบิตต่อ 1 วินาทีไปจนถึง 100 ล้านบิตต่อ 1 วินาที ซึ่งสูงกว่าที่เราใช้กันบนสายโทรศัพท์กว่า 1000 เท่า ทางออกทางนี้จึงเป็นทางออกที่เปิดไปสู่โลกอีกแบบหนึ่งได้ด้วย ทางออกนี้จึงน่าจะเป็นทางออกที่ดีที่สุดในขณะนี้

Connectivity ในสายตาของยูสเซอร์

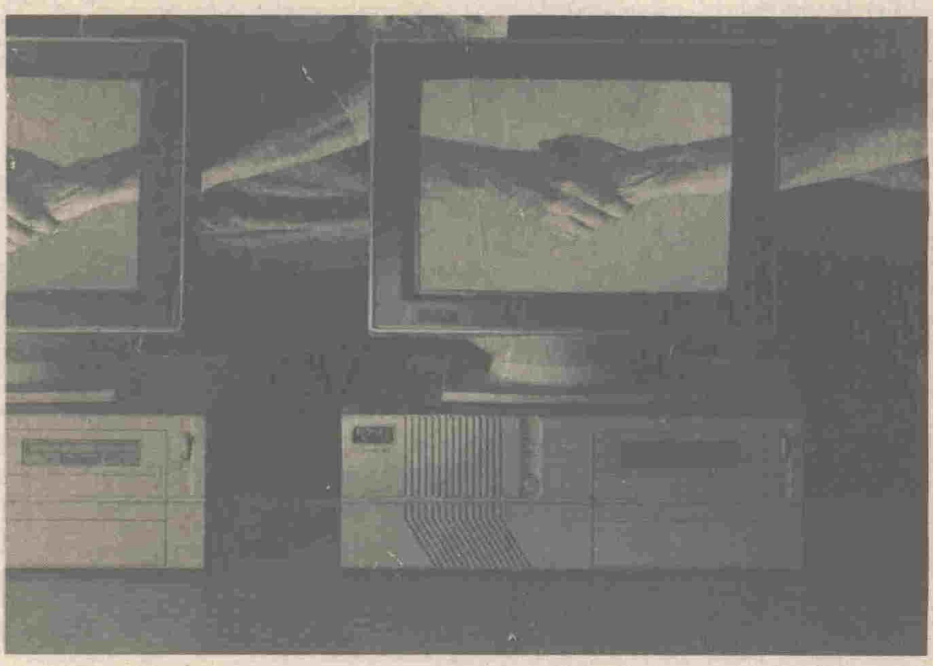
ที่กล่าวมาทั้งหมดนั้นท่านจะเห็นว่า ผมเน้นในเรื่องของการที่จะทำให้โปรแกรม 2 โปรแกรม ซึ่งวิ่งอยู่บนเครื่องคอมพิวเตอร์คนละเครื่อง สามารถติดต่อสื่อสารและแลกเปลี่ยนข้อมูลกันได้ ความสามารถอันนี้จำเป็นสำหรับโปรแกรมเมอร์ที่จะมาพัฒนาระบบโปรแกรมให้เราใช้ แต่ผู้ใช้ (User) หลาย ๆ คนเขาจะไม่สนใจในเรื่องนี้เลย คำว่า "Connectivity" ในสายตาของพวกเขาจะหมายถึงความสามารถที่จะให้ระบบงานที่เขามีอยู่แล้ว 2 ระบบงานได้เรียกหาข้อมูลของกันและกัน ยกตัวอย่างเช่น เขามีระบบการทำรายงานเรื่องสภาวะเศรษฐกิจของประเทศอยู่ในมือ และเขารู้ว่าในฝ่ายแลกเปลี่ยนเงินตราต่างประเทศก็มีระบบการทำรายงานการไหลของเงินอยู่ เขาต้องการที่จะดึงข้อมูลจากเพื่อนพนักงานในฝ่ายแลกเปลี่ยนเงินตราต่างประเทศมาออกเพิ่มในรายงานของเขา หรือในบางทีก็อาจเป็นเรื่องที่ง่ายกว่านั้น คือในรายงานของเขานั้นต้องแสดงตัวเลขจากฝ่ายแลกเปลี่ยนเงินตราต่างประเทศอยู่แล้ว แต่ระบบของเขาต้องไปขอข้อมูลจากฝ่ายแลกเปลี่ยนเงินตราต่างประเทศมา แล้วเขาต้องคีย์ข้อมูลเหล่านั้นเข้าไปในระบบของเขา เขาจึงอยากจะให้ 2 เครื่องนี้โยงถึงกัน แล้วให้เครื่องหยิบข้อมูลส่งข้ามมาให้โดยอัตโนมัติ

ครับ "Connectivity" ในสายตาของยูสเซอร์นี้มันก้าวลึกไปกว่าความต้องการของโปรแกรมเมอร์อีกหลายโค้งและสร้างปัญหาได้แยะจริง ๆ ทั้งนี้เพราะระบบงานเดิมที่เขามีอยู่นั้นส่วนใหญ่จะอยู่ในไมโครคอมพิวเตอร์ที่ใช้ "ดอส" (DOS) เป็นโอเปอเรติงซิสเต็มส์ และโปรแกรมที่ช่วยในการทำรายงานนั้นมักจะเป็นสเปรดชีต (LOTUS 1-2-3) หรือไม่ก็เป็นโปรแกรมที่เขียนขึ้นเองด้วยภาษาดีเบส (dBASE) การเชื่อมโยงในสายตาของพวกเขาดูจะง่ายนิดเดียว แต่เมื่อถามฝ่ายคอมพิวเตอร์แล้วจะได้รับคำตอบที่ไม่เป็นที่สบอารมณ์ เช่น บางคนบอกว่า "อีแค่คีย์ข้อมูลซ้ำนิดเดียวก็มาบ่นอยู่ได้" บางคนก็บอกว่า "ยากครับ, ยากมาก ๆ เพราะโลตัสมันอ่านข้อมูลของดีเบสไม่ได้ และโลตัสจะอ่านข้อมูลข้ามไปหาโลตัสอีกตัวหนึ่งไม่ได้ คุณอาจต้องเอา 2 งานนี้มารวมกันอยู่บนเครื่องและโปรแกรมเดียวกัน ฯลฯ" เมื่อเจอคำตอบแบบนี้เขาก็ไปเสาะหาที่พึ่งทางอื่น ทางหนึ่งคือถามเพื่อน อีกทางหนึ่งคือถามบริษัท เรื่องนี้เราจะพบว่าเพื่อนจะช่วยย้ำว่าความต้องการนี้สมควรได้รับการพิจารณา ทั้งนี้เพราะเพื่อนก็อยากได้อยู่เหมือนกัน ส่วนคำตอบจากบริษัทนั้นจะทำให้เขาตื่นเต้น คือ บริษัทจะ

บอกว่า "ทำได้ แต่คุณต้องเปลี่ยนเครื่องเป็นยูนิกซ์เสียก่อน"

บริษัทที่ขายเครื่องประเภทริสค์ (RISC) และขายเครื่องประเภทไฮเพาเวอร์เวอร์กสเตชันจะถือเป็นโอกาสอันดีที่จะมาดึงลูกค้าประเภทนี้ พวกเขาได้รับการสั่งสอนมาว่าเครื่องของเขามี "Connectivity" สูง เขาเชื่อสนิทใจว่าระบบงาน 2 ระบบงานสามารถเชื่อมโยงและแลกเปลี่ยนข้อมูลกันได้ แต่เมื่อพูดถึงรายละเอียดว่างานนี้จะทำได้อย่างไร พวกเขาก็จะขอเวลาศึกษาก่อน (แต่ในความเป็นจริงแล้วเขาไม่มีเวลาและภูมิปัญญาที่จะศึกษาเรื่องนี้ได้) เมื่อยูสเซอร์เวียนไปถามก็จะได้รับคำตอบเดิม คือ "ทำได้แน่นอน แต่รายละเอียดมันเยอะ ผมคิดว่าสั่งซื้อแล้วผมจะส่งซัพพอร์ตไปช่วยดูให้" แต่อย่างไรก็ตาม การตัดสินใจนั้นไม่ใช่เรื่องง่าย ทั้งนี้เพราะเครื่องและซอฟต์แวร์ชนิดยูนิกซ์นี้แพงขึ้นอีกหลายแสนบาท แถมยังไม่แน่ใจว่าจะวิ่งโปรแกรมโลตัสหรือดีเบสที่ตนมีอยู่ได้หรือไม่ และถึงแม้ว่าจะวิ่งได้ก็ยังไม่รู้ว่าจะให้ใครแก้ไขโปรแกรมให้เกิด "Connectivity" กันได้

ครับ ผมเองก็ไม่อยากจะพูดว่าระบบยูนิกซ์นั้นทำไม่ได้ หรือทำได้ยากกว่าระบบดอส ทั้งนี้เพราะระบบดอสที่เรามีอยู่นั้นจะให้เอามาเชื่อมโยงกันจะทำได้ยากมาก ๆ ดอสสามารถเชื่อมโยงโดยผ่านช่องสื่อสาร "COM1" และ "COM2" ได้ แต่การแก้ไขโปรแกรมโลตัสหรือดีเบสให้มีความสามารถส่งข้อมูลผ่านช่องสื่อสารดังกล่าวนั้นเป็นเรื่อง "เกือบเป็นไปไม่ได้" ทั้งนี้เพราะโลตัสและดีเบสเป็นโปรแกรมสำเร็จรูป ซึ่งเขาขายให้เรามาในรูปแบบของภาษาเครื่อง (Object Code) พวกเราไม่เคยมีประสบการณ์ในการแก้ไขโปรแกรมในรูปแบบดังกล่าวนี้ ใครทำเรื่องนี้ได้ก็น่าจะติดต่อบริษัททั้งสองเพื่อขายความคิดไปเลย อาจรวยเป็นมหาเศรษฐีได้เร็วกว่ามานั่งทำงานเป็นโปรแกรมเมอร์อยู่ในเมืองไทย



สรุป

โดยสรุปแล้ว ถ้ายูสเซอร์ต้องการให้มีการเชื่อมโยงในระดับโปรแกรมงาน (Application) ผมก็คิดว่า ยูนิกซ์น่าจะเป็นหนทางออกที่ดีที่สุด แต่ท่านก็คงต้องใช้เวลาศึกษาและทดลองกันพอสมควร เช่น อาจต้องใช้วิธีสั่งก๊อบปี้ข้อมูลจากเครื่องของฝ่ายแลกเปลี่ยนเงินตราฯ มาก่อน แล้วต้องใช้คำสั่ง (Command) ในระบบยูนิกซ์เพื่อจัดการกับข้อมูลนั้นได้แล้วก็ย้ายลงไปเป็นอินพุตไฟล์ที่เคยใช้รองรับข้อมูลที่คีย์เข้าไป แล้วจึงจะวิ่งโปรแกรมออกรายงานกันต่อไป เรื่องนี้จะเหมาะสมหรือไม่ก็ต้องไปคำนวณดู ท่านอาจต้องเสียเงินเพิ่ม 2-3 แสนบาท แล้วท่านก็จะได้ความสะดวกสบายบางอย่าง เช่น ไม่ต้องคีย์ข้อมูลเข้าไปใหม่ เรียกดูข้อมูลของคนอื่นได้ แต่อย่างไรก็ตาม การคำนวณ "Cost/Benefit" นี้ ต้องละเอียดถี่ถ้วนหน่อยนะครับ เช่น เมื่อท่านเชื่อมโยงเครื่องของท่านเข้ากับเครื่องของคนอื่นแล้วก็ต้องมีความเสี่ยงในด้านการถูกแอบดูข้อมูล แอบแก้ไขข้อมูล ฯลฯ ท่านอาจต้องเสียเวลาและแรงงานในการสร้างเกราะป้องกันตนอีกมาก ค่าใช้จ่ายเหล่านี้ต้องนำมาคำนวณด้วย

สรุปแล้ว "Connectivity" นั้นเป็นดาบ 2 คม คมหนึ่งนั้นมีประโยชน์มันจะทำให้เราทำอะไร ๆ ได้อีกมาก แต่อีกด้านหนึ่งนั้นเป็นเรื่องของเวลาแรงงาน เงิน และความเสี่ยงที่เราต้องแบกรับ เรื่องนี้ในขณะนี้อาจดูแล้วก้ำกึ่งกัน คือประโยชน์ก็มีพอสมควร แต่ค่าใช้จ่ายก็สูงพอสมควร แต่อย่างไรก็ตาม ในอนาคตนั้นประโยชน์ดูจะสูงขึ้นเรื่อย ๆ ดังนั้นเราจึงเตรียมคนเอาไว้แต่เนิ่น ๆ